AFRL-IF-RS-TR-2002-106
Final Technical Report
May 2002

INDEPENDENT TESTING AND INTEGRATION OF THE NETWORK MANAGEMENT GUARD PROXY

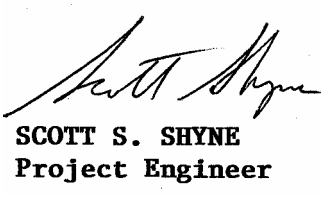
BAE Systems Portal Solutions, Inc.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

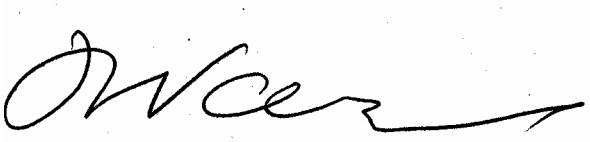
AFRL-IF-RS-TR-2002-106 has been reviewed and is approved for publication.

APPROVED:



SCOTT S. SHYNE
Project Engineer

FOR THE DIRECTOR:



WARREN H. DEBANY, Jr., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 02	Final Apr 99 – Jan 02

4. TITLE AND SUBTITLE

INDEPENDENT TESTING AND INTEGRATION OF THE NETWORK MANAGEMENT GUARD PROXY

5. FUNDING NUMBERS

C - F30602-99-C-0077
PE - 63789F
PR - 233G
TA - 02
WU - 01

6. AUTHOR(S)

Adam Hovak and Joe Riolo

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

BAE Systems Portal Solutions, Inc.
111 East Chestnut Street
Rome, NY 13440

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFGA
525 Brooks Road
Rome, NY 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2002-106

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Scott S. Shyne, IFGA, 315-330-4819, shynes@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

The document provides an overview of the network management guard/proxy. The network management guard/proxy is a modified firewall that sits between two networks of different classification levels. This modified firewall is used to provide network monitoring and control of multilevel networks. It sits between the high-side network management platform and the low-side network management platform via a virtual private network (VPN). The guard/proxy only allows information to pass between the two network management platforms based upon the integrity policy defined at the guard/proxy startup. This allows the high network management system to be able to monitor both the high-side and low-side networks. Furthermore this document contains information on the configuration of the boundary device, demonstrations that were accomplished, and testing that was completed during this contract.

14. SUBJECT TERMS

Boundary device, network management system, guard/proxy, firewall, network classification, multilevel security, network monitoring, network control, virtual private network (VPN)

15. NUMBER OF PAGES

54

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

List of Exhibits

List of Tables

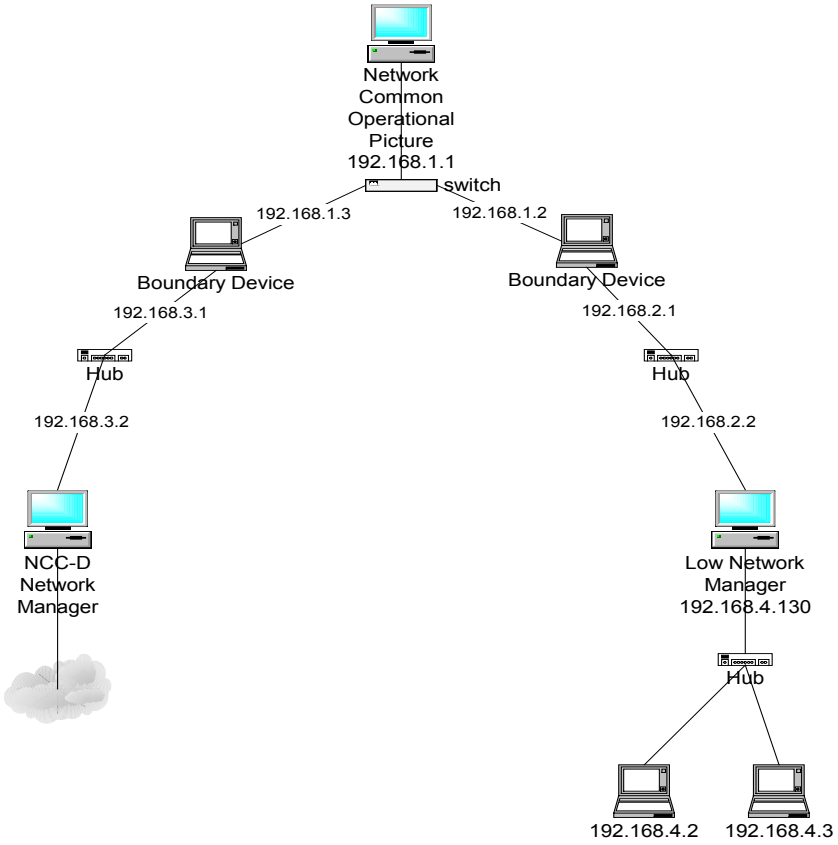
1.0 INTRODUCTION

This document describes the work accomplished under the Independent Testing and Integration of the Network Management Guard Proxy effort, Contract No. F30602-99-C-0077 for the Air Force Research Laboratory. This report includes the value of the network management guard/proxy to multi-level networks.

2.0 PROJECT DESCRIPTION

The network management guard/proxy is a modified firewall that sits between two networks of different classification levels. This modified firewall is used to provide network monitoring and control of multi-level networks. It sits between the high-side network management platform and the low-side network management platform via a virtual private network (VPN). The guard/proxy only allows information to pass between the two network management platforms based upon the integrity policy defined at the guard/proxy startup. This allows the high network management system to monitor both the high-side and low-side networks.

Exhibit 1. The Original MDNM Architecture



This architecture works as follows. At the top of the architecture is the network Common Operational Picture (COP) which displays multiple domains in one picture. The COP uses Simple Network Management Protocol (SNMP) to query the Management Information Base (MIB) data of the low network manager. The MIB data contains up-to-date information about the resources that they manage. The data found at the low Network Management System (NMS) are replicated and placed into the COP's database. This data contains status information of the systems managed by the low NMS.

A unique design of the boundary device is the ability to span Multi-Level Security (MLS) domains. This is permissible because of our careful evaluation of each packet that tries to pass through the MDNM boundary device. When a packet arrives at the boundary device it is first met by the firewall. The firewall allows or denies the packet depending on origination, destination, and protocol. For instance, if a user on the COP wants to telnet to the low NMS, the firewall configuration on the boundary device wouldn't allow this to occur because it only allows SNMP and Internet Control Message Protocol (ICMP).

The second stage of the boundary device is the proxy. The proxy is code that was developed by ARCA Systems. This code takes each individual packet, verifies that the SNMP command is allowed, and then repackages the packet to ensure anonymity as well as command integrity.

Anonymity is important so that the low NMS is not given any information about the COP. Command integrity is tested to ensure the validity of the command being sent based on the rules configured in /etc/SnmpProxy.cfg. For example, an SNMP GetRequest is denied from the low NMS to the COP but a GetRequest is allowed from the COP to the low NMS. Flow control is set this way because the COP holds information from multiple domains and this information cannot be exchanged between the low domains.

BAE SYSTEMS tested the guard/proxy based on real world scenarios, thereby examining its usefulness to network managers. One of the tasks within this study included integration of the guard/proxy into a real-world, multi-level network. The main task involved testing the guard/proxy by using tools that included HP OpenView 5.0 and other commercial management platforms based on SNMP.

3.0 BOUNDARY DEVICE

This section describes how to install the software necessary for the MDNM boundary device. First we will discuss the installation of the operating system, FreeBSD, then the X server. Next we will discuss installing PCMCIA cards with specific steps for those with Xircom cards. The last step in setting up a boundary device involves modifying system files. There are two possible ways to do this, both of which will be described.

3.1 INSTALLATION OF FREEBSD

The following section describes how to install FreeBSD 3.2 onto the boundary device.

Begin by setting your bios to boot off the CD-ROM and then proceed with booting off of the FreeBSD 3.2 CD-ROM. A Graphical User Interface (GUI) installation screen will appear. Scroll down to **Novice** installation. We chose a novice installation to ensure then every installation is the same. Select OK to go to an MS-DOS style fdisk.

Press **d** to delete the used partitions.
Press **a** to use the entire disk. It will ask if you want to do this with a true partition. **[yes]**

Press **q** to finish.

Next choose your boot type. Choose **standard.**

Now create BSD partitions inside the fdisk partitions. Select **[ok]** to continue.
The following commands will create the correct size partitions on the Dell Inspiron 5000s.

[c] [2000M] [FS] [/]
[c] [1000M] [Swap]
[c] [10000M] [FS] [/var]
[c] [2000M] [FS] [/usr]
[c] [2296M] [FS] [/root]

When you are done press **q** to finish.

On the next screen you will be asked what distribution to install. Choose **X-developer (No. 2)**

Next you will be asked if you wish to install DES cryptographic software. **[yes]**
Check **des** **Basic DES encryption services**
Choose **exit**

Would you like to install FreeBSD ports collection? **[yes]**
Select components you need from Xfree86.
Select **[exit]** Select **[exit]**

Choose installation media. **[CDROM]**

Would you like to continue with installation? **[yes]**
Now FreeBSD will be installed. This will take approximately 20 minutes.

You will now be prompted with numerous post installation questions. The following is a list of the appropriate answers.

Would you like to configure any Ethernet or SLIP/PPP network devices? **[no]**
Will this machine be an IP gateway? **[no]**
Do you want to allow anonymous ftp connections to this machine? **[no]**
Do you want to configure this machine as an NFS Server? **[no]**
Do you want to configure this machine as an NFS client? **[no]**
Would you like to customize your system console settings? **[no]**
Would you like to set this machine's time zone now? **[yes]**
Is this machine's CMOS clock set to UTC? **[no]**
Select **[America – North and South]**
Select **[United States]**
Select **[Eastern Time]**
Does the abbreviation 'EDT' look reasonable? [**yes]**
Does the system have a mouse attached to it? [**yes]**
Select **[select port]**
Select **[PS/2]**
Select **[enable]**
Is it working? Move the mouse. If it is working, type **[yes]**
Select **[exit]**
Would you like to configure your x-server at this time? **[no]**
FreeBSD package collection. Would you like to browse the collection? **[no]**

Would you like to add any initial user accounts? **[no]**
Enter the system manager's password. **[ok]**
jefx02 [return]
jefx02 [return]
Visit the general configuration menu for a chance to set any last options? **[no]**
Select **[Exit Install]**

Now you are at a command prompt. The previous steps set you up with a working version of the operating system. The following steps will configure the X server.

3.2 CONFIGURING AN X SERVER

The next important step in setting up a boundary device is configuring X windows. X windows contains a series of servers that allows the video card to interact with the desktop environment. Without the appropriate X server, we would not be able to use the BSD KDE.

The Dell Inspiron 5000 has a video card that is not supported by the FreeBSD 3.2 X Window package so we need to install a newer version. As of this writing the newest X Window package is 4.01. We will use this package for the boundary device.

The X Window package we need for the boundary device can be downloaded at ftp.xfree86.org/pub/Xfree86/4.0.1/binaries/FreeBSD-3.x**.** The files need to be downloaded into **/usr/X11R6.**

A list of the necessary files follows.

Xinstall.sh	**extract**	**Xbin.tgz**	**Xlib.tgz**
Xman.tgz	**Xdoc.tgz**	**Xfnts.tgz**	**Xfenc.tgz**
Xetc.tgz	**Xvar.tgz**	**Xxserv.tgz**	**Xmod.tgz**

Once the files have been downloaded they must be installed. To do so, move into the correct directory by typing **cd /usr/X11R6**. Next run the install program by typing **sh Xinstall.sh**. This program will prompt you with the following questions.

Do you wish to continue? **[yes]**

The next few prompts will ask if you want to overwrite config files. Answer **[yes]** for all.

Do you wish to have the new links installed? (Links for the termcap file) **[no]**

Do you wish to have the rstart link installed? **[no]**

Do you want to move files? **[yes]**

The installation file has now finished. Now we need to do a few things to enable the X server to run properly. First, move into the appropriate directory by typing **cd /usr/X11R6/bin**. Next run the Xfree86 program by typing **./Xfree86 –configure**. This command creates a default configuration file for the X server. Our next task is to move the file into its correct location. You can do this by typing **cd /root** to move into the correct directory, and then **cp XF86Config.new /etc/X11/XF86Config** to copy the file.

Xfree86 is now installed. The default window manager is Open Windows. If you want to run the KDE you need to put the CD back in the drive. Then you need to add the KDE package. This is done by typing **cd /stand** to move into the correct directory, and **./sysinstall** to start up the install shell.

The following are the options you need to select to install KDE.
Select **[configure]**
Select **[Packages]**
Select **[CDROM]**
Select **[kde]**
Select **[kde-1.1.1]**
Select **[Cancel]**
Select **[install]**

You will be prompted if you want to continue. **[yes]**

After KDE installs select **[cancel]** and then **[exit Install]**

When you exit the install program, a command prompt will appear. Move to the root directory by typing **cd/root**. Now add the word **kde** to the file .xinitrc by typing the command **echo kde >> .xinitrc**

The computer is now successfully set up with FreeBSD 3.2 and Xfree86 4.01.

3.3 CONFIGURING PCMCIA CARDS

This section describes how to install two Xircom PCMCIA cards in FreeBSD as well as Windows NT. The system on which we installed the Xircom cards was a Dell Inspiron 3500. The laptop we used contained a Pentium II 400 MHz processor, 128 MB of RAM, and a 4.5 GB hard drive. We segmented the hard drive into two partitions, a 2 GB partition with Windows NT and a 2.5 GB partition with FreeBSD. We were limited to a 2 GB partition for Windows because we chose to remain with the FAT 16-file system.

3.3.1 XIRCOM INSTALLATION UNDER WINDOWS NT

On the Windows partition we installed Windows NT 4.0 with Service Pack 5. After a successful installation of Windows NT we attempted to install two similar 3com PCMCIA cards. Only one card would work due to a limitation in Windows NT. After some research it was discovered that Windows NT 4.0 does not support two PCMCIA cards of the same model. FreeBSD also does not support these model 3com cards so we removed them from NT. FreeBSD does not support these PCMCIA cards because they are cardbus and there are no drivers yet written for cardbus cards in FreeBSD. We decided to use the two Xircom cards that were not of the cardbus type. Due to the Window NT limitation described above both Xircom cards will not work simultaneously in Windows NT so we installed only one. If a second card is inserted the machine becomes confused because it cannot decide with which card to negotiate. If a second card is installed the machine locks up at the login screen. A KMODE_EXCEPTION is thrown which results in a complete lockup. The only solution to the lockup is to restart with the last known good configuration. A single Xircom card worked in Windows NT with the following settings.

I/O Port	0x280
Memory Address	0xd4000
Interrupt	3
Mode	I/O

Interrupt 3 is a serial port that is not used at this time. The serial port in the BIOS was disabled to make IRQ 3 available.

After reviewing a list of supported cards in FreeBSD, the 3com 3c589 series PCMCIA cards have better support in FreeBSD. We found extra hardware and discovered that the 3com cards have better support under FreeBSD 3.2.

3.3.2 XIRCOM INSTALLATION UNDER FREEBSD

First attempt at Xircom card installation (unsuccessful)
We installed PAO version 3.2 after numerous recommendations from FreeBSD newsgroups. PAO enables FreeBSD to drive many PCMCIA (PC Card) cards (Ethernet, Wireless LAN, FAX/Modem, ISDN, Digital Cellular, SCSI, Flash ATA, ATA HDD, and ATAPI CD-ROM) and provides a "hotplug" function for your PC Card on your laptops running FreeBSD. The "hotplug" function automatically configures a PC Card when it is inserted.

After what seemed to be a successful installation of PAO, we found that several errors existed. The network light was present on the network dongle (the cable connecting network card to RJ45 jack) but the card could not be reached through ping. This indicated that physically the connection was sound but the card could not negotiate with an appropriate IRQ. All available IRQs were tried with no success.

At this point, Scott Mitchell was contacted. He is the keeper of the Xircom drivers for FreeBSD. He claimed that the software PAO is buggy and provides another opportunity for error. On his suggestion PAO was uninstalled and the Xircom drivers were downloaded from http://ukug.uk.FreeBSD.org/~scott/xe_drv/. Version 1.20 (1999/06/13) was installed. The process of uninstalling PAO was not completely successful. PAO manipulated a program called pccardd present in FreeBSD. Pccardd is a program that recognizes the PCMCIA cards and loads the appropriate drivers necessary to use the cards. When PAO was removed pccardd was not replaced with its pre-PAO version. This became a problem when the xe drivers were installed because they used the original pccardd, which was no longer there. Since we did not want to be left with any residual effects of PAO, the FreeBSD partition was deleted, the partition was formatted, and the Xircom card installation was started anew.

Second attempt at Xircom card installation (successful)

We began the second attempt at installing the Xircom cards with a clean version of FreeBSD. The machine had just been formatted and we were starting from the beginning. In this attempt we would install the xe drivers we downloaded earlier. The attempt proved successful. The following is a step-by-step description of how to correctly install the drivers.

3.3.2.1 INSTALL THE XE DRIVERS AND MAKE THEM RECOGNIZABLE TO THE KERNEL

Unzip and untar xe_drv.tar.gz. If you are unfamiliar with this, type **tar -xzvf xe_drv.tar.gz**. This command does two things. It unzips the file and then it extracts the compressed file. You will find three files if_xe.c, if_xereg.h, and README. Copy if_xe.c and if_xereg.h into /sys/i386/isa.

Add the following line to usr/src/sys/i386/conf/files.i386. It makes the driver available to the kernel.

i386/isa/if_xe.c optional xe device-driver

3.3.2.2 ADDITIONAL FILE MODIFICATIONS

a) Modify the kernel configuration file in /usr/src/sys/i386/conf

The original kernel configuration file is named GENERIC. Save a copy of the original working kernel. In this example copy it to the name PCMCIA by typing **cp GENERIC PCMCIA**. You can name the kernel file with any name you wish by substituting a name for PCMCIA. In this report, the PCMCIA will be referred to as your kernel configuration file. All of our editing will occur in the file PCMCIA.

We need to make additional modifications to the kernel in order to free up available IRQs. Xircom support stated that the Xircom CE3B generally likes IRQs 3, 9, 10, and 11. We must therefore "comment out" or disable unused devices claiming these IRQs. Comment out the following line to disable /dev/mcd0 providing it is not being used. To determine if it is being used, read the device output when the machine first boots. All devices are listed; the used ones have configurations set.

#device mcd0 at isa? port 0x300 bio irq 10

Add the following lines to your kernel configuration file. (Note: Some of these may already be there).

```
controller      card0
device          pcic0   at      card?
device          pcic1   at      card?
device          xe0     at      isa? port? net irq ?    #Makes xe0 an available device
device          xe1     at      isa? port? net irq ?    #Makes xe1 an available device
```

You can now explicitly specify an I/O port address for the device through the XE_IOBASE kernel option. If present, this will be picked up by the CEM56/REM56 support code and used as the base address for the Ethernet part of the card. You only need to care about this if a) you have a CEM56 or REM56 and b) the driver chooses a port address that conflicts with some other device on your system (most likely at 0x300). In this case, specify an address that definitely isn't used by anything else, for example,

```
options         XE_IOBASE=0x320
```

After the line

```
device xe1      at isa? port? net irq?
```

you will need to modify the next four lines to read as follows.

```
device sio0     at isa? disable port "IO_COM1" flags 0x10 tty irq 4
device sio1     at isa? disable port "IO_COM2" tty irq 3
device sio2     at isa? disable port "IO_COM3" tty irq 5
device sio3     at isa? disable port "IO_COM4" tty irq 9
```

The word “disable” has been added to these lines. These four lines disable the COM ports that seem to be taking up available IRQs. Once again make sure these COM ports are not being used by the system already. Once these are disabled, IRQs 3, 4, 5, and 9 will be available to set up the Xircom CE3B.

Comment out the following lines to disable extra interfaces. The Xircom card uses only xe0 and xe1. Leaving interfaces that conflict with xe0 and xe1 will result in an incomplete kernel compile.

```
#device ed0 at isa? port 0x280 net irq 10 iomem 0xd8000
#device ie0 at isa? port 0x300 net irq 10 iomem 0xd0000
#device ep0 at isa? port 0x300 net irq 10
#device ex0 at isa? port? net irq?
#device fe0 at isa? port 0x300 net irq ?
#device le0 at isa? port 0x300 net irq 5 iomem 0xd0000
#device lnc0 at isa? port 0x280 net irq 10 drq 0
#device ze0 at isa? port 0x300 net irq 10 iomem 0xd8000
#device zp0 at isa? port 0x300 net irq 10 iomem 0xd8000
#device cs0 at isa? port 0x300 net irq ?
```

Lastly the Berkeley packet filter may be uncommented because it is a useful debugging tool. This step is optional. If you wish to run tcpdump you will need to follow this step. Here is an example.

pseudo-device bpfilter 4 #Berkeley packet filter

Build and install your new kernel. "Your kernel's name" refers to the renamed configuration file. Above we named it PCMCIA. Substitute whatever name you used above.

config "your kernel's name"
cd ../../compile/"you kernel's name"
make depend
make
make install

b) Edit /etc/pccard.conf.

This file holds the specifications of the Xircom cards. The xe driver will attempt to establish a link when the interface is brought up by ifconfig. This will happen automatically if your pccard.conf has an "insert" line like the one below. By default it will attempt to auto-detect the media type, on both 100Mbit and 10Mbit networks. The "media" option to ifconfig can be used to force a particular media type if the auto selection does not work. Your /etc/pccard.conf should be similar to:

```
#Xircom CE3 CreditCard Ethernet 10/100
card "Xircom" "CreditCard 10/100"                  #matches pccardc dumpcis output
config 0x1 "xe0" 4                                 #set interface xe0 to IRQ 4
config 0x1 "xe1" 9                                 #set interface xe1 to IRQ 9
insert echo Xircom CreditCard Ethernet inserted    #echo card inserted
insert /etc/pccard_ether xe0                       #run driver
remove echo Xircom CreditCard Ethernet removed     #echo card removed
remove /etc/pccard_ether xe0 delete                #run driver
```

NOTE: The strings in the card entry need to match the CIS of your particular card. If in doubt, run "pccardc dumpcis" and check the value of tuple 0x15 (version 1 info). You need the manufacturer and card version strings. In the config line, make sure you choose an IRQ that isn't used by anything else on the system.

NOTE: The "0x1" in the "config" line above refers to a particular configuration supported by your card. You will need to check the "pccardc dumpcis" output to make sure you are using a valid configuration. Look for tuples of type "0x1b (Configuration entry)." For example, a typical CE3B card should produce something similar to:

```
Tuple #8, code = 0x1b (Configuration entry), length = 25
   000:  c1 c1 bd 7f 55 4d 5d 3e 46 46 06 e0 17 17 e4 60
   010:  00 00 0f 70 bc 8e 10 00 20
         Config index = 0x1(default)
         Interface byte = 0xc1 (I/O)  +RDY/-BSY active, wait signal supported
```

Vcc pwr:
Nominal operating supply voltage: 5 x 1V
Minimum operating supply voltage: 4.5 x 1V
Maximum operating supply voltage: 5.5 x 1V
Continuous supply current: 3.5 x 100mA
Max current average over 1 second: 4 x 100mA
Max current average over 10 ms: 4 x 100mA
Power down supply current: 1 x 100mA
Wait scale Speed = 1.2 x 10 ms
RDY/BSY scale Speed = 1.2 x 10 ms
Card decodes 23 address lines
IRQ modes:
IRQs: NMI IOCK BERR 0 1 2 4 8 9 10 12
Memory space length = 0x60
Max twin cards = 0
Misc attr:
Tuple #9, code = 0x21 (Functional ID), length = 2
000: 06 00
Network/LAN adapter
Tuple #10, code = 0x22 (Functional EXT), length = 8
000: 04 06 00 80 c7 dd 3f 74
Voice services available:
Tuple #11, code = 0x22 (Functional EXT), length = 2
000: 01 02
Modem interface capabilities:
Tuple #12, code = 0x22 (Functional EXT), length = 5
000: 02 80 96 98 00
Data modem services available:
Tuple #13, code = 0x22 (Functional EXT), length = 5
000: 02 00 e1 f5 05
Data modem services available:
Tuple #14, code = 0x22 (Functional EXT), length = 2
000: 03 01
Tuple #15, code = 0x22 (Functional EXT), length = 2
000: 03 03
Tuple #16, code = 0x22 (Functional EXT), length = 2
000: 05 00
Modem interface capabilities:
Tuple #17, code = 0xff (Terminator), length = 0

There are five config entries here with indices 0x27, 0x1f, 0x17, 0xf, and 0x3f. You should try each of these values in the "config" line of your pccard.conf until you find one that works with your particular setup.

NOTE: If either of the CIS strings are longer than 30 characters (as with some Intel and Xircom CEM cards), you will exercise a bug in pccardd. To fix it you will need to recompile your

pccardc and pccardd after changing the definition of CIS_MAXSTR from 30 to 254 in /usr/src/usr.sbin/pccard/pccardd/readcis.h. As far as I known, no one has patched this in-stable yet.

c) Modify /etc/rc.conf to look similar to the following.
Some of the lines may already be there. This file is called upon boot to enable the PC cards, recognize the necessary interfaces, and set the appropriate ipaddress and netmask.

moused_port="/dev/psm0"	#sets a device for the mouse
moused_enable="YES"	#enables the use of the mouse
hostname="killians"	#sets hostname(choose anything you like)
pccard_enable="YES"	#enable the PC Card
network_interfaces="xe0 xe1 lo0"	#sets up interfaces (Xircom uses xe0,xe1)
ifconfig_lo0="inet 127.0.0.1"	#set the loopback ipaddress

NOTE: /etc/rc.conf overrides the file /etc/defaults/rc.conf. It is possible to look in /etc/defaults/rc.conf to see available overrides but it is suggested that you do not edit this file directly.

d) Modify /etc/rc.local
This file modification is a quick fix to a loading order error. The rc.conf is designed to set ipaddress and netmask for the network devices xe0 and xe1. The error occurs in the order in which the xe_driver and pccardd are called. When rc.conf is called it cannot see the interfaces because the driver has not yet been called. Add the following two lines to rc.local with your network addresses.

```
ifconfig xe0 inet 192.168.102.62
ifconfig xe1 inet 192.168.102.61
```

rc.local is called after everything else is loaded in bootup including the xe driver. After the driver has been run these two lines set the network addresses of the two interfaces.

e) Add a line to /etc/rc
The file etc/rc is essentially the autoexec.bat of the FreeBSD system. This file gives the operating system the order in which to do things when the system is being started. The only thing we have to add to this file is a line that makes the computer sleep. Sleep is necessary for this setup because the network cards need time to be configured. If the system does not sleep then files such as /etc/rc.conf fail because the network cards have not been recognized by the system. We need to add the two lines to this file right after

```
#start up PC-Card configuration
if [ -f /etc/rc.pccard ]; then
        . /etc/rc.pccard
fi
```

The lines we need to add are:
echo Sleeping to allow for configuration of PC Cards
sleep 25

The first line will echo the message, "Sleeping to allow for configuration of PC Cards." The second line actually makes the system sleep for 25 seconds. We chose 25 seconds after trying numerous lengths of time. This amount of time allows the PC Cards to be correctly configured before the system continues.

f) Reboot
You should get an "xe0 not found" message in the normal boot output. The card will be probed and attached when pccardd is run.

3.3.3 FAMILIAR ERRORS AND POSSIBLE SOLUTIONS

3.3.3.1 BIOS PROBLEM

Error read as follows.
The amount of system memory has changed or the suspend to disk file is absent. See suspend to disk file in the online user guide for instructions.

I had to run PHDisk provided by Dell to create a suspend to disk file. PHDisk can be found on the Dell utilities disk included with the machine. The suspend to disk file is destroyed every time the computer boots into FreeBSD. To access BIOS, boot to A:\PHDisk (create suspend to disk file). Reboot and BIOS is accessible. Setting the suspend to disk file to suspend to RAM seems to be a fix for this problem.

3.3.3.2 DEBUG ERROR

Error kept occurring.
Feb 9 12:25:37 killians /kernel: xe0: media_status

This is not an error. This is a debugging statement. To turn this message off you could either look for the line in if_xe.c that prints the offending message, and comment it out, or to remove all of the debug messages from xe, delete these three lines towards the top of the file.
#ifndef XE_DEBUG
#define XE_DEBUG 1 /* Increase for more voluminous output! */
#endif

Recompile the kernel and the error is gone.

3.3.3.3 DRIVER ALLOCATION FAILED FOR XIRCOM

This usually means that the IRQ set in the pccard.conf is already allocated to something else. Change the IRQ to something more acceptable (i.e., 3, 4, 9, 10, 11).

Troubleshooting

I found it helpful to isolate the laptop on a small network (1 hub, 2 machines). Run a packet sniffer on the other end and see if packets are actually being sent and received. FreeBSD has a packet sniffer included. To enable the sniffer you have to make a kernel config. The last line in your kernel file needs to be uncommented. When correctly configured, the line should read:

pseudo-device bpfilter 4 #Berkeley packet filter

After editing the above line recompile the kernel. Refer to section 3.3.2.2 to read more about kernel compiling.

To use the sniffer type **tcpdump** –i xe0. This will set the sniffer to interface xe0. More information on tcpdump is available in the manpage.

The sniffer was used to see exactly where packets were being sent and where they were being received. If you find packets are not received on the interface bpf0 (tcpdump) most likely you have an IRQ conflict.

3.3.3.4 WATCHDOG TIMEOUT

Watchdog timeout usually occurs when the card does not recognize the IRQ you specified in the pccard.conf. If you find the card continually rebooting there is an IRQ conflict.

NOTE: When you set configurations such as **Ifconfig xe0 inet 192.168.102.62** the card will have a watchdog timeout. The card resets in order to set the ipaddress. Watchdog timeout is not necessarily bad.

3.4 INSTALLATION SCRIPT

Prior to JEFX, the task of setting up a boundary device was very tedious. The installation became much easier with the introduction of an installation script, which takes most of the manual work out of a boundary device installation. The script is described below.

The installation script, called install.sh, is located on an installation disk labeled **boundary device install**. This disk contains the following 10 files necessary for installation:

.login	libpcap.so.2
.xinitrc	libpcap-0.4.tar
codelib.tar.gz	low2high
high2low	README
install.sh	syslog.conf

These files can change since there is still active coding within the project. As long as the newest versions of these files are placed onto the installation disk, the installation process should work fine.

In order for this installation disk to work, there are a few requirements that need to be met. Before running the installation script, be sure the following criteria are met.

- ✔ The system is running Freebsd 3.2.
- ✔ There are two working network interfaces cards with IP addresses.
- ✔ There is a running version of Xwindows.
- ✔ There is root login

After you have met the above criteria, you need to edit your kernel to add a few capabilities that the boundary device software requires. To edit the kernels, carefully follow these steps.

1) Edit your kernel file

The default file is /usr/src/sys/i386/conf/GENERIC. If you have made edits, use the most current file. Add the following options:

```
options     IPFIREWALL              #firewall
options     IPFIREWALL_VERBOSE      #print dropped packet info
options     IPDIVERT                #divert sockets
```

After adding these options change the last line of the kernel to read

```
pseudo-device     bpfilter 8     #Berkeley packet filter
```

2) Compile your kernel

Once you have made all the changes, you need to compile your kernel for the changes to take effect. To do this, move to the correct directory by typing **cd /usr/src/sys/i386/conf**. Now type the following commands to compile and install the kernel.

config <kernel filename>
cd ../../compile/<kernel filename>
make depend
make
make install

Now the kernel is compiled. In order for the changes to take effect, reboot the machine. Once the machine has come back up, run the installation script. To do so, insert the disk labeled **boundary device install** into the 3.5 floppy drive and follow these steps.

1) Run the script
Mount the installation disk at point /mnt. Type **mount_msdos /dev/fda0 /mnt** cd to /mnt. Type **[sh install.sh] [low interface] [high interface]**

2) Finish editing the files
Check the /etc/SnmpProxy.cfg file. Add the MAC addresses and IP addresses of both the high NMSs and the low NMSs.

3) Edit /etc/hosts
Be sure to include the ip addresses of the high NMS, low NMS, highsidebd, and lowsidebd.

After the above three steps have been completed, all the software for the boundary device should be installed and the boundary device should now be in working order.

3.5 WITHOUT THE INSTALLATION SCRIPT

This section describes how to configure the guard proxy boundary device without an installation script. For information on setting up a laptop in FreeBSD and configuring the interfaces see the previous sections of this document. Follow the steps below to manually set up a boundary device.

3.5.1 EDIT /ETC/RC.FIREWALL

/etc/rc.firewall is the file that sets the rules for the firewall. There are a few rules that must be added before the boundary device will work correctly.

$fwcmd add 100 pass all from any to any via lo0
$fwcmd add 200 deny all from any to 127.0.0.0/8
$fwcmd add divert 6969 udp from lownms to lowsidebd via xe0
$fwcmd add divert 9696 udp from highnms to lownms via xe1
**$fwcmd add divert 6969 icmp from lownms to lowsidebd **

via xe0 icmptype 0,18
**$fwcmd add divert 9696 icmp from highnms to lownms **
via xe1 icmptype 8,17
$fwcmd add allow icmp from highsidebd to highnms via xe1 icmptype 0,18
$fwcmd add allow icmp from highnms to highsidebd via xe1 icmptype 8,17
$fwcmd add 65000 deny log all from any to any

The above lines are rules that specify what type of information can travel through the interfaces. Please notice in every rule there is an interface associated with it whether it is xe0, xe1, or any. It is important when setting up these rules to know that xe0 is the device connected to the lowside and xe1 is the device connected to the highside. If the network cards in your system are different than xe0 and xe1, please substitute your device names in these places.

3.5.2 ADD A LINE TO /ETC/SYSLOG.CONF

There is only one small modification that needs to be made to /etc/syslog.conf. On the first line after the comments add the line:
local1.* /var/log/proxy.log

The rest of the file should contain the following lines. If they are not present, add them.

mail.info /var/log/maillog
lpr.info /var/log/lpd-errs
cron.* /var/cron/log
!startslip /var/log/slip.log
!ppp

3.5.3 MODIFY /ETC/RC.LOCAL

The file rc.local is a file that is called after every other file in the boot sequence. The modification we need to make to this file is very simple. Add the following two lines.

/high2low
/low2high

3.5.4 MODIFY /ETC/HOSTS

The /etc/hosts file is a file that relates hostnames to IP addresses. The basic file structure consists of two columns, the network address column and the hostname column. The left column represents the network address of the hostname in the right column. The five basic addresses that must to be in this file are:

- ✔ loopback
- ✔ highnms
- ✔ highsidebd
- ✔ lowsidebd
- ✔ lownms.

First is the loopback. Most commonly the address is 127.0.0.1 with a hostname of localhost. The line in /etc/hosts should look similar to the following:

127.0.0.1 localhost

The following entries are the ones unique to the network management boundary device. On the boundary device we are creating there are two network cards each with its own unique IP address. These network cards will be referred to as lowsidebd and highsidebd. We referred to the network devices above in /etc/rc.firewall as xe0 and xe1 where xe0 is connected to the lowside and xe1 is connected to the highside network. We will refer to those interfaces again in /etc/hosts. Set /etc/hosts to reflect the IP addresses of the network devices on your machine, for example:

175.75.75.2 highsidebd #IP address of highside network device
192.168.4.2 lowsidebd #IP address of lowside network divice

The last two entries in the /etc/hosts file represent the network interface of the management system looking over a domain. Remember that the boundary device sits between two network management systems. Each of these network management systems has two network interfaces. One interface goes to the domain it oversees and one goes to the boundary device. The COP is referred to as high NMS and the network management system overlooking the lowside network is referred to as low NMS. In the /etc/hosts file you need to put the network address of the high NMS and the low NMS so the boundary device can reach the management systems. For example the next two lines of the /etc/hosts file should look similar to the following.

175.75.75.5 highnms
192.168.4.131 lownms

Please note the boundary device is in both network domains.

3.5.5 ADD LINES TO /ETC/RC.CONF

The file /etc/rc.conf is a file that is called upon at boot to set up network interfaces, router configurations, or firewall status. It is important to add and edit a few lines in this file to get the Network Management Guard proxy to work.

The lines you need to edit in /etc/rc.conf are the lines that set the IP address of the network interfaces. You need to substitute the new hostname we set above in /etc/hosts for the IP address. After the substitution the lines should look similar to the following.

ifconfig_xe0=”inet lowsidebd netmask 255.255.255.0”
ifconfig_xe1=”inet highsidebd netmask 255.255.0.0”

The last lines you need to add to this file regard router and firewall setup. The following lines should be added to /etc/rc.conf.

gateway_enable=”YES”	#enables this machine as a gateway
defaultrouter=”NO”	#this is not a router by default
router_enable=”NO”	#this is not a router
firewall_enable=”YES”	#enable the firewall
firewall_type=”Proxy”	#the firewall type is Proxy
firewall_quiet=”NO”	#the firewall will not act in quiet mode
inetd_enable=”NO”	#do not enable inetd
portmap_enable=”NO”	#do not enable portmap

The file /etc/rc.conf is now correctly configured. Save the file and exit the editor.

3.5.6 CREATE /VAR/LOG/PROXY.LOG

In this step we will create proxy.log. To create a file in FreeBSD the command is **touch**. To create /var/log/proxy.log we type **touch /var/log/proxy.log**. proxy.log is a log file of all network traffic this machine has observed.

3.5.7 EDIT /ROOT/.LOGIN AND /ROOT/.XINITRC

There are two files that we have to edit in this step. Both files are related in some way so we will do them in one step. When you are running /bin/csh the file .login is essentially a batch file that performs certain operations once you log in. The lines you need to make sure you have in .login are:

set path = (/sbin /bin /usr/sbin /usr/bin /usr/X11R6/bin /usr/local/bin /usr/local/sbin) #set path
setenv MANPATH “/usr/share/man:/usr/local/man” #path for manpages
umask 22 #set file creation mode mask
startx #start xwindows

The next file edit we need is related to the .login file. As you can see above in .login, the last call we made was to startx which starts Windows. The file .xinitrc is a file that controls what is started along with Windows. For the boundary device we want to be able to view what is going on between the interfaces. To do this we need some terminals and windows to open when Windows starts. The following lines create x-terminals.

xterm -geometry 164x17+0+100 -rv -sb -sl 1000 -fg white &

sleep 1

xterm -geometry 164x17+0+0 -T "<----- Low Network [xl0]" -rv -sb -sl 1000 -fg green -e tcpdump -i xe0 &

xterm -geometry 164x17+0+255 -T "High Network [pn0] ----->" -rv -sb -sl 1000 -fg red -e tcpdump -i xe1 &

xterm -geometry 164x17+0+510 -T "SNMP Filter Output" -rv -sb -sl 1000 -fg gold -e tail -f /var/log/proxy.log &

The last line we need to add to .xinitrc deals with the window manager. For the network management boundary device we want the kde window manager so we add the following as the last line to .xinitrc.

kde

3.5.8 COPY COMPRESSED FILES TO /ROOT/PROXY AND /ROOT/PROXY/PROXY

To be able to complete this section you must have two files, **codelib.tar.gz** and **proxy3.3.tgz**. Once you have these files, copy proxy3.3.tgz to /root/proxy. To do this you will first need to create the proxy directory by typing:

cd /root [return] #to get to the root directory
mkdir proxy [return] #creates directory called proxy

From the directory where the floppy drive is mounted, copy proxy3.3.tgz to /root/proxy by typing **cp proxy3.3.tgz /root/proxy [return]**. The file is now in /root/proxy. The next thing you need to do is unzip and uncompress the file. To do this make sure you are in the correct directory. Type **cd /root/proxy [return]**. At this point unzip and uncompress proxy3.3.tgz by typing **tar –xzvf proxy3.3.tgz [return]**. proxy3.3.tgz has now been extracted into the necessary directories.

The next file that must be unzipped and uncompressed is codelib.tar.gz. First, copy the file into the correct directory before uncompressing it. From the directory where the floppy drive is mounted, copy codelib.tar.gz to /root/proxy/proxy by typing **cp codelib.tar.gz /root/proxy/proxy [return]**. The file has been copied. The next thing to do is unzip and

uncompress the file. To do this make sure you are in the correct directory. Type **cd /root/proxy/proxy**. From there type **tar –xzvf codelib.tar.gz** to unzip and uncompress the file.

3.5.9 COPY LIBPCAP.SO.2 TO /USR/LIB

In this step you must have the file libpcap.so.2. Once you have this file copy it to /usr/lib. Before doing so, however, make a copy of the original libpcap.so.2 in /usr/lib by typing **cd /usr/lib [return]** to move to the appropriate directory and **cp libpcap.so.2 libpcap.so.2.ORG** to rename the file. Now that the original file has been renamed, go to the directory where the floppy drive is mounted and type **cp libpcap.so.2 /usr/lib**. This copies the file into /usr/lib.

3.5.10 EDIT THE KERNEL

The kernel configuration file must be edited. To do so go to the directory where the kernel configuration file is located. Type **cd /usr/src/sys/i386/conf**. This command brings you into the correct directory. The first step in a kernel modification is to save the previous kernel file by renaming the kernel and using the renamed version. If you have never modified your kernel configuration file the file you need is GENERIC. If you have previously modified your kernel configuration file, choose the file you last edited for this step. Rename your file by typing **cp GENERIC GUARDPROXY**. This step creates a copy of the old kernel that can be edited without the worry of modifying the original kernel. Please note that you will only make modifications to the new file GUARDPROXY. Do not edit GENERIC; save that file in case you need it later. You have to do only two minor edits to the kernel file. First, add some options and then make the Berkeley packet filter active. The options you need to add come from a file called LINT located in the same directory as your kernel config file. In LINT you will find many options that can be added to the kernel. The options you need to add are:

```
options     IPFIREWALL              #firewall
options     IPFIREWALL_VERBOSE      #print dropped packet info
options     IPDIVERT                #divert sockets
```

These options add firewall capabilities to the kernel.

The next thing you need to do is make the Berkeley packet filter active. Change the last line in the kernel config file to

```
pseudo-device  bpfilter 8     #Berkeley packet filter
```

Note: We have uncommented the Berkeley packet filter to make it active and increased the number available to eight.

Now that all of the editing is done, save the file and exit. Compile the kernel by making sure you are still in the directory where your kernel configuration file is located and by typing the

following lines. "your kernel config's name" refers to the kernel configuration file you edited above. We used GUARDPROXY.

config "your kernel config's name"
cd ../../compile/"your kernel config's name"
make depend
make
make install

Your kernel is now correctly compiled. You must restart before this kernel is loaded.

3.5.11 CREATE THE DEVICES FOR THE BERKELEY PACKET FILTER

To be able to use the eight Berkeley packet filters created above, create devices for each one. To do sofind the /dev directory. Type **cd /dev [return]** to move to the device directory. From there create all eight devices by typing:

./MAKEDEV bpf0 [return]
./MAKEDEV bpf1 [return]
./MAKEDEV bpf2 [return]
./MAKEDEV bpf3 [return]
./MAKEDEV bpf4 [return]
./MAKEDEV bpf5 [return]
./MAKEDEV bpf6 [return]
./MAKEDEV bpf7 [return]

All eight devices have been created and they are ready to be used.

3.5.12 COPY SOURCE CODE INTO /USR/LOCAL/SRC

To complete this step you need the file libpcap-0.4.tar.gz. Once you have the file copy it to the correct directory and then uncompress it. First copy the file into the correct directory. From the directory where the floppy drive is mounted, copy libpcap-0.4.tar.gz to /usr/local/src by typing:
cp libpcap-0.4.tar.gz /usr/local/src [return]

Now that the file has been copied, uncompress it. Move to the correct directory and uncompress it by typing:

cd /usr/local/src [return] #moves to /usr/local/src
tar –xzvf libpcap-0.4.tar [return] #uncompresses file

Now that the appropriate files have been unzipped and uncompressed you can move a couple of files. After following the above directions you are in /usr/local/src. Move to /root/proxy/proxy/pcapmods by typing **cd /root/proxy/proxy/pcapmods [return]**. From this directory copy pcap-bpf-0.4a3.c to /usr/local/src/libpcap-0.4 and rename it as **pcap-bpf.c** by typing:
cp pcap-bpf-0.4a3.c /usr/local/src/libpcap-0.4/pcap-bpf.c [return].

Now that the file has been copied to the appropriate directory, create the object file by moving to /usr/local/src/libpcap-0.4 by typing:
cd /usr/local/src/libpcap-0.4 [return]

Configure the directory to be compiled by typing:
./configure [return]
make [return]
make install [return]

3.5.13 MAKE AND MAKE INSTALL THE NEWLY COPIED FILES

In order for the copied files to work, you need to make and install them. This could not have been done earlier because all of the files were not in the correct directories. First change directories to /root/proxy/proxy/h2l by typing:
cd /root/proxy/proxy/h2l [return]

Next, remove the old object files by typing:
make clean [return]

Then create the new object files by typing:
make [return]
make install [return]

From here change directories again by typing:
cd .. [return]
cd l2h [return]

You should now be in /root/proxy/proxy/l2h. Once again remove the object files by typing:
make clean [return]

Then create the new object files by typing:
make [return]
make install [return]

3.5.14 EDIT SNMPPROXY.CFG

In /root/proxy/proxy you will find a file called SnmpProxy that must be copied and edited in order for the boundary device to work. First, copy the file from /root/proxy/proxy to /etc by typing:

cd /etc #to move to /etc
cp /root/proxy/proxy/SnmpProxy.cfg . #copy SnmpProxy to /etc

Now that the file is in its place make sure it contains the correct information. This file contains specific interface information like device names, MAC address, IP address, and divert port. An example of /etc/SmnpProxy.cfg would look similar to the following.

LowDevice: xe0
LowMAC 00:80:c7:dd:3f:74
LowIP 192.168.4.2
LowDivertPort 6969
HighDevice xe1
HighMAC 00:80:c7:f9:2a:74
HighIP 175.75.75.2
HighDivertPort 9696

If you are not sure of the information above there is a way to check. To obtain a listing of the configured devices type **ifconfig –a [return]**. In this output you will see information ordered by device. Underneath each device listing you will find the MAC and IP addresses. Once you add the above lines to /etc/SnmpProxy.cfg you need to add two more lines. These lines specify the MAC and the IP addresses of the HighNMS and LowNMS. The lines should look similar to the following.

HighNMS: 08:00:20:9c:4c:4c 175.75.75.5
LowNMS 08:00:20:9c:4e:05 192.168.2.131

Please note that the HighNMS and LowNMS are referred to in the /etc/hosts file. If you are unsure of the IP address, view the /etc/hosts file to get the correct addresses.

All of the required information has now been added to /etc/SnmpProxy.cfg. The last thing that must be done is to make sure that everything below the line
LowNMS 08:00:20:9c:4e:05 192.168.2.131
is set to allow. Once this has been done, save the file and exit.

If you followed the above steps, the boundary device is correctly set up.

3.6 CHANGING AN IP ADDRESS

In many situations you will find that you need to change an IP address of an interface on the boundary device. You may have to do this to one or both interfaces. In any case, you will probably need to change an interface address at some point. There are numerous files that need to be changed in order to successfully change an interface IP address. Among these files are **hosts, SnmpProxy.cfg**, **rc.conf, and S72inetsvc**.

3.6.1 EDIT /ETC/HOSTS

/etc/hosts is the file that contains a list of host names and associative IP addresses. Make sure that the IP address you wish to change is changed here first. Use your favorite editor to open the file and change the appropriate address.

3.6.2 EDIT /ETC/SNMPPROXY.CFG

/etc/SnmpProxy is a file that contains both IP and MAC addresses. If you are only trying to change the IP address, you need not consider the MAC address. Only when you are adding an interface must you include the new MAC address. Use you favorite editor to open this file and change the relevant IP address.

3.6.3 /ETC/RC.CONF

/etc/rc.conf is a setup file called by the operating system upon boot. rc.conf is the file that sets the IP address to the specific interface. If you followed the correct steps above, you have hostnames in /etc/rc.conf so nothing needs to be edited. If you did not create the host file correctly, then you need to do so.

3.6.4 ALLOWING FTP THROUGH THE BOUNDARY DEVICE

Occasionally it is necessary to transfer programs or files from one management station to another. If the files are too large to transfer on a disk, then another method must be used. A viable alternative is opening up the boundary device for ftp traffic. Please note that when the boundary device is opened for ftp traffic, it no longer serves its purpose as a firewall.

The first step in opening the boundary device is disconnecting it from the managed domain. This means that you must disconnect the low NMS from its domain before you open up the boundary

device. This is not only a safety precaution but also a necessity. The managed domain may find the central NMS, which could lead to unnecessary arp requests.

The next step in setting up the boundary device is setting the appropriate firewall rules to allow ftp. To add the appropriate rule to the boundary device type:
ipfw add 1 allow ip from any to any

Please note that this completely opens the boundary device. This rule allows all traffic to pass through the boundary device.

Now that the boundary device is open you can ftp any files you might need. To close the firewall type:
ipfw delete 1

This command deletes the rule we created to open up the boundary device.

If all of the above steps were completed successfully, you should have a working boundary device.

4.0 NETWORK MANAGEMENT SYSTEM

This section is a technical description of how to configure the network management system that sits between the boundary device and the COP. The NMS is a machine that overlooks a domain. HP OpenView is used to manage the domain and provide information to the COP on a specified basis.

The NMS is currently running on Solaris 2.6. We chose to install the software on a Sun Ultra 60. The computers came with 18-gigabyte drives partitioned into many smaller slices.

A significant amount of space was left for the /var partition. This was done so there is ample room for storing the log files.

4.1 OPERATING SYSTEM INSTALLATION

The first thing we did with the NMSs was to install the operating system, Solaris 2.6. The partitions were set up as follows.

/	**2 gigabytes**
swap	**500 MB**
/usr	**2769 MB**

/var	**4 gigabytes**
/opt	**6 gigabytes**
/ust/openwin	**2 gigabytes**

After a successful installation we chose the Common Desktop Environment for our X Windows manager.

4.2 PROGRAM INSTALLATIONS

After the installation there are many programs that must be installed in order for HP OpenView to work. The first piece of software that must be installed is Netscape Navigator 6.0 or higher.

Once Netscape is installed, there are still four patches to the Solaris 2.6 architecture to install. The first patch that must be installed is the thread patch for Java runtime. The patch number is 105181-20. The second patch that must be installed is 105284-33. The third patch is 107733-06. This third patch obsoletes patch number 105490-07. The last patch that must be installed is 105210-27 for error code exit status. These patches can be found at http://sunsolve.sun.com.

An important program that might be needed to unzip the downloaded files is **gzip**. Most of the patches that are downloaded are both tarred and gzipped. This program can be downloaded from http://www.sunfreeware.com.

After all the patches are installed you can continue with the installation of HP OpenView. The default install will allow your system to manage 250 nodes for a period of 60 days. You must register your IP address with HP to get a permanent license number.

If all of the above installs occurred successfully, you will have a system that only needs a few minor tweaks before it is operational.

4.2.1 CONFIGURATION OF THE LOW NMS

Low NMS stands for low network management system. A low NMS is a management system running HP OpenView that monitors a domain. The lownmses are managed by a central NMS. This section will describe the file modifications necessary for the successful integration of a low NMS.

A few files need to be created and edited for a NMS to work correctly. For instance, you have to create the second interface and set it to the correct IP address.

4.2.1.1 /ETC/HOSTNAME.$#%^

The next file to be created is **/etc/hostname.$#%^**. This file contains the hostname of the interface where $#%^ is the interface name. In most situations on Solaris the hostname will be hme* where * is an integer starting at 0. When you first install Solaris, the operating system will recognize one interface and set this interface as default hme0. On the low NMSs you need to configure two interfaces, **hostname.hme0** and **hostname.hme1**. Create the file hostname.hme1 and add a hostname to that file. For example, on a low NMS with a hostname of bass the file hostname.hme0 will look like

bass #interface hme0

while the file hostname.hme1 will look like

bass2 #interface hme1

4.2.1.2 /ETC/HOSTS

This file contains both the hostnames and the IP addresses of the interfaces you just created. This file typically has two columns, the first being the IP address and the second being the hostname. On a low NMS there are only two entries you need in the host file. Examples of these entries follow.

155.244.30.145 bass #(hme0)
129.132.37.41 bass2 #(hme1)

4.2.1.3 /ETC/RESOLV.CONF

It is necessary to create this file on the low NMS if the domain that the low NMS is managing has a nameserver. If the domain does not have a nameserver, then skip this step. An example of what this file might look like is

nameserver 128.132.1.201

4.2.1.4 /ETC/DEFAULTROUTER

It is necessary to create this file on the low NMS if the managed domain has a default router. If the domain does not have a router, then this file does not need to be created. An example of what this file might look like is
155.244.30.1

4.2.1.5 /ETC/NSSWITCH.CONF

This file is necessary to edit on the low NMS if the managed domain has a dns. If the domain does have a dns, then you need to add the word dns to a line in nsswitch.conf. The line you need to edit looks like

hosts: files

In order for the machine to look to the dns for hostnames you need to change the line to look like

hosts: files dns

4.2.1.6 HP OPENVIEW CONFIGURATION

The following instructions should be followed after HP OpenView has been installed on the low side NMS machine.

1. Make sure HP OV NNM is not running. If it is running, exit all OVW sessions and issue the following command:

   $OV_BIN/ovstop

2. Make sure the SNMP master agent is not running by executing the following command:

   ps –ef | grep snmpdm
   kill –9 process id #

3. Edit the set community name line in the /etc/snmpd.conf file to read as follows.

   set-community-name: name #enter community name
   where, name is the community name of your choice.

   For example:

   get-community-name: public
   set-community-name: netmon
   trap-dest: 175.75.75.5 #only on the lowNMS

   The first three lines must be added in order for HP OpenView to work. The IP address of **trap-dest** should be the IP address of the high NMS.

4. Restart the SNMP master agent by executing the following commands:

   snmpd *(full pathname /usr/sbin/snmpd*)

5. Start NNM by executing the following command.

   $OV_BIN/ovstart –v

6. Make sure the ovtopmd process is running by executing the following command:

   $OV_BIN/ovstatust ovtopmd

Note: The preceding instructions were taken from *HP OpenView: A Guide to Scalability and Distribution for Network Node Manager*, April 1996, pages 4-17 and 4-18.

4.2.2 CONFIGURATION OF CENTRAL NMS

Central NMS stands for central Network Management System. The central NMS is the management system running HP OpenView that monitors all the low NMSs. The central NMS is commonly referred to as the COP because it provides a picture of all managed domains. This section will describe the file modifications necessary for the successful integration of the central NMS.

A few files need to be created and edited for the NMS to work correctly.

4.2.2.1 /ETC/HOSTS

The first file to be edited is /**etc/hosts**. This file needs to be added to the central NMS but it is not necessary to add all of the detail of this file to the low NMSs. It is this file that contains all the IP addresses of the managed systems. The file is generally made up of two columns, the first containing the IP address and the second the hostname. It is possible to add more than one name by adding another column. An example of the host file on the central NMS is as follows.

#High Network that connects boundary devices to the Central NMS
175.75.75.5 harps.high.com COP-NMS COP
175.75.75.2 RS-BD herb
175.75.75.3 LS-BD

#Low Network NMSs
129.132.37.40 bass
192.168.4.130 saranac

In the first section (High Network) you will notice the first IP address is the COP or the central NMS. All entries below the central NMS are connected to the NMS via a switch. In the second section (Low Network) you will notice the IP addresses of the low NMSs. In the above example we have two low NMSs, each managing their own domain. If another low NMS is added to the architecture, add a line containing its IP address and hostname.

4.2.2.2 HP OPENVIEW CONFIGURATION FOR THE COP

The following instructions should be followed after HP OpenView has been installed on the central NMS (high side NMS).

1. Default route set to: /etc/defaultrouter [should set static route for better security]

175.75.75.1 *(high side of Firewall)*

2. Create Netseed file: /etc/opt/OV/share/conf/netseed:

175.75.75.5 *(mulberry)*
192.168.4.220 *(maple)*

3. Modify netmon.lrf file: /etc/opt/OV/share/lrf/netmon.lrf so the last 2 lines read:

   Ovs_YES_START:ovtopmd,pmd,ovwdb:-p -s
   /etc/opt/OV/share/conf/netseed:Ovs_WELL_BEHAVED:15:

3(a). After modifying netmon.lrf run the following series of commands.

   $OV_BIN/ovaddobj /usr/OV/lrf/netmon.lrf
   $OV_BIN/ovstop netmon
   $OV/BIN/ovstart netmon

4. Reboot machine.

5. After reboot, login.

6. Start OVW in one of the cmdtool windows: $OV_BIN/ovw &.

Note: The preceding instructions were taken from *HP OpenView: A Guide to Scalability and Distribution for Network Node Manager*, April 1996, pages 4-19 and 4-20.

4.2.2.3 CONFIGURING A MANAGEMENT STATION FOR A GIVEN COLLECTION STATION

1. Make sure the NNM background processes are running by executing the following command.

   $OV_BIN/ovstatus

   If the NNM background processes are not running, execute the following command.

   $OV_BIN/ovstart

2. Determine which collection stations already exist by executing the following command.

   $OV_BIN/xnmtopoconf -print

   This will give you a list of current collection stations and tell you if they are managed or unmanaged. If the collection station does not exist, continue to the next step. If the collection station does exist, continue with step 4.

3. Add the collection station to the topology by executing the following command:

   $OV_BIN/xnmtopoconf –add –unmanage *collectionstation*

4. Configure the set community name to be used when communicating with the collection station by selecting Options:SNMP Configuration from the OVW menu bar. This will bring up the SNMP Configuration dialog box.

5. In the SNMP Parameters pane of the dialog box, specify in the Set Community text field the same set community name that you configured on the collection station.

6. Click [OK] to close the SNMP Configuration dialog box.

7. To make sure the collection station is configured correctly, execute the following command.

   $OV_BIN/xnmtopoconf –test

8. Manage the collection station by executing the following command:

   $OV_BIN/xnmtopoconf –manage *collectionstation*

4.2.2.4 CONFIGURING THE AUTOMATIC DATABASE SYNCHRONIZATION

[This is a workaround for an apparent bug in HP OV NNM demo version]

1. Determine which collection stations already exist by executing the following command.

   $OV_BIN/xnmtopoconf -print

   This will give you a list of current collection stations and tell you if they are managed or unmanaged. If the collection station does not exist or is unmanaged, go to section 4.2.2.3**,** else continue. If the collection station does exist, continue with step 3.

2. Verify that Collection Station or low side NMS(maple) STATUS is “normal.”

3. Verify that communications and configuration are still valid by executing the following command.

   $OV_BIN/nmdemandpoll –s *collectionstation*

   Verify that output shows, "**Synchronization with station *collectionstation* complete**" is displayed.

4. As a workaround for a current OV problem create the following script: *(example is given using the vi editor)*

   vi $OV_BIN/OvdbSync.scpt
   I *<- insert command for vi editor; Type following verbatim in first column (i.e., no tab before each entry)*

   #!/bin/csh

   while 1

   echo "Starting Synchronization…"
./nmdemandpoll –s *collectionstation* **#Name of low NMS/CollectionStation**
   sleep 90 **#time in seconds between polls**

   end

:wq! *<- write and quit command for vi editor.*

9. Change attribute of OVdbSync.scpt to be executable by executing the following command.

   chmod 775 OVdbSync.scpt

10. Execute the script you just created by issuing the following command.

$OV_BIN/OVdbSync.scpt *(add '**&**' if you want in background)*

4.2.2.5 /OPT/OV/BIN/OVDBSYNC.SCPT

This file syncs the low NMSs to the central NMS and must be copied into **/opt/OV/bin** after it is created.

4.2.2.6 /ETC/HOSTNAME.HME0

For the central NMS you only need to have hostname.hme0. It is this interface that is connected to a switch. Also connected to the switch is a boundary device for every low NMS present.

4.2.2.7 CONFIGURING ROUTING

Routing is a very crucial part of the boundary device because the boundary device was designed for routing. Some configurations must be set up so that packets can make it through the boundary device and into the network management systems.

4.2.2.8 EDIT /ETC/RC2.D/S72INETSVC

This file resides on the network management systems. The only station on which you need to make this change is the COP. The COP is the network management system that overlooks all the other network management system. There are two lines that need to be added to /etc/rc2.d/S72inetsvc. Those lines are:

/usr/sbin/route add 192.168.4.131 175.75.75.2
/usr/sbin/route add 129.132.37.41 175.75.75.3

These two lines allow communication between the low network management systems (low NMS) and the high network management systems (high NMS). For example, the numbers **192.168.4.131** and **129.132.37.41** represent the high side of the lownmses. There are two lines above because there are two lownmses and subsequently two managed domains. You need a line for every management station you have, not including the COP. The numbers **175.75.75.2** and **175.75.75.3** represent the high side interface of the boundary devices. The high side interface of the boundary device is the one that is connected to the COP via the switch. See Exhibit 1.

If all of the above steps were completed successfully, you should have working network management systems.

5.0 TESTING

The goal of this project has been to test the boundary device in numerous situations, record test results, and report to the developers. In order for this to work correctly, we designed a testing process as well as a test plan. This section will describe the details of both the testing process and the test plan.

5.1 TESTING PROCESS

This section describes the necessary configurations that must be made for a useful testing process to take place.

One of the first problems that becomes evident when trying to test a boundary device is the quantity of messages that are being sent and received. HP OpenView is set to poll nodes and update topologies every few minutes. With all this traffic going by, it is difficult to see output caused by the test.

In this test we will be trying to send SNMP data from the central management station through the boundary device to a low management station. In order for the receiving machine to accept data and return data appropriate read and write communities must be set in the snmpd.conf file.

The way to reduce network traffic is to stop HP OpenView from repetitiously querying the network management systems. All the changes need to be made to the version of HP OpenView running on the central management station (otherwise known as the COP).

STEP 1

Go into the HP OpenView menu. Under the **options** menu select **SNMP Configuration**. This will bring up a window. About midway down the window you will see a row of information called default status polling. The value will most likely be set to 5m. You need to select that row and change the polling to 24h. This will tell HP OpenView to only query the management stations once a day.

STEP 2

Go into the HP OpenView menu. Under the **options** menu select **Network Polling**. This will bring up a new window. At the top of the window is a drop-down list. Select the drop-down list and choose the option **General**. In the General menu window you need to turn off two options, **topology checks** and **perform Configuration checks**, by unchecking the boxes next to the option.

In the same window, select the drop-down list once again and choose the option **ipDiscovery**. In the ipDiscovery window deselect the checkbox for **Discover new IP node**.

With the same window open, select the drop-down list once again and choose the option **Status polling**. In the Status polling window deselect the checkbox for **perform status polling**.

STEP 3

The next step in reducing message traffic is managing the stations. Make sure that the central management station is no longer managing the low network management system. The low network management system that you will be querying through the boundary device will have to be set to **unmanaged** while testing takes place. To do this, go to a free command prompt on the central NMS and type **xnmptopoconf –print**. This will give you a list of all the management stations that are being managed. The low network management system that you are querying should be listed there. To unmanage the station type **xnmtopoconf –unmanage willow.net** where willow.net is the name of the low network management system.

STEP 4

This is the last step before testing can begin. First, make sure that **OvdbSync.scpt** is not running. The easiest way to do this is to reboot. When the machine comes back up, login and start HPOV with the command **ovw &**. Do not synchronize the data bases.

The above steps set up the central management station for testing the boundary device. By default, the boundary device has all snmp traffic open. You need to turn off all the snmp traffic except the traffic for which you would like to test. The file to do this is **/etc/SnmpProxy.cfg**.

The SnmpProxy.cfg file looks similar to the following.

```
#
#Protocol to be processed
#Omissions cause the type to deny
#
SNMPv1:      allow
SNMPv2c:     allow

#
#Omissions cause the type to deny
#
H2LSet:              allow
H2LGet:              allow
H2LGetNext:          allow
H2LGetBulk:          allow
H2LInform:           allow
H2LTrap:             allow
H2LResponses:        allow

#
#Omissions cause the type to deny
#
L2Hset:              allow
L2Hget:              allow
L2HgetNext:          allow
L2HgetBulk:          allow
L2Hinform:           allow
L2Htrap:             allow
L2Hresponses:        allow
```

There are three sections to this file. The first section specifies the SNMP version. The second section allows or disallows traffic from the central network management system to the low network management system. The last section allows or disallows traffic from the low network management system to the central management station.

To test an SNMP daemon you need to enable a minimum of three things. The first is the version, SNMPv1, or SNMPv2c. The second thing is the daemon you want to test. You need to choose whether you are testing the high to low traffic or the low to high traffic. Once you choose which direction the daemon will be allowed to travel, you need to turn on the response for the opposite direction. This allows the response to return to the program successfully. The configuration of **/etc/SnmpProxy.cfg** where we tested SNMPv2c GET follows.

#Protocol to be processed
#Omissions cause the type to deny
#
#SNMPv1: allow
SNMPv2c: allow

#
#Omissions cause the type to deny
#
#H2LSet: allow
H2LGet: allow
#H2LGetNext: allow
#H2LGetBulk: allow
#H2LInform: allow
#H2LTrap: allow
#H2LResponses: allow

#
#Omissions cause the type to deny
#
#L2Hset: allow
#L2Hget: allow
#L2HgetNext: allow
#L2HgetBulk: allow
#L2Hinform: allow
#L2Htrap: allow
L2Hresponses: allow

Notice the three lines that are allowed (uncommented lines denote allowed lines). Also note that every time a configuration is made to the SnmpProxy.cfg, the computer needs to be rebooted before the settings will take effect.

The program we used to test the SNMP daemons was a program written by Mr. Joe Riolo. The program is in the root directory of all the machines in the NDF. The file is called **snmptester.ui.tcl**. The program is written in tcl/tk so tcl/tk needs to be installed before you try to run this program. Type the file name at the command prompt to get the program to start. Once the program is started you need to choose a few options. First you need to enter the **host** that you are going to query. The next is the **set** and **get communities.** After that you need to select a daemon from one of two columns. The first column represents the available daemons

for SNMPv1. The second column represents the daemons available in SNMPv2c. After you choose a daemon you are nearly ready to start your testing. That last thing you need to do is set up **snoop** on the interface that you are trying to query. In the NDF this interface is hme0 on the central NMS and hme1 on the low NMS. To activate the snoop command you need to type **snoop –d hme0**. This listens on interface hme0 for network traffic.

After setting up the snooping device on the low NMS you can systematically test each daemon, one by one, by turning them on or off in the SnmpProxy.cfg.

5.1.1 AFTER THE TESTING

After you are satisfied with the tests that you have run, you need to place the machines in the condition in which you found them. To do this, follow the steps below.

STEP 1

Go into the HP OpenView menu. Under the options menu select **SNMP Configuration**. This will bring up a window. About midway down the window you will see a row of information called default status polling. The value will be set to 1d or 24h. You need to select that row and change the polling to 5m. This will set it so that HP OpenView queries the management stations every five minutes.

STEP 2

Go into the HP OpenView menu. Under the options menu select **Network Polling**. This will bring up a new window. At the top of the window is a drop-down list. Select the drop-down list and choose the option **General**. In the General menu window, change two options, **topology checks** and **perform Configuration checks**, by selecting the checkbox next to the two options to turn them on.

In the same window, select the drop-down list once again and choose the option **ipDiscovery**. In the ipDiscovery window reselect the checkbox for **Discover new IP node**.

With the same widow open, select the drop-down list once again and choose the option **Status polling**. In the Status polling window reselect the checkbox for **perform status polling**.

STEP 3

Now make sure the low NMS is once again managed. First type **xnmptopoconf –print**. This will give you a list of all the management stations that are being managed as well as the unmanaged stations. The station that is unmanaged is the one we turned off earlier. To turn management back on for this station type **xnmtopoconf –manage willow.net** where willow.net is the name of the low network management system.

STEP 4

Be sure that before you finish the testing process you put the boundary device back to normal by modifying /etc/SnmpProxy.conf. This file should be set so it only allows the appropriate network management traffic.

STEP 5

This is the last step before you are back to normal. Type **reboot**. When the machine comes back up it should be in the same state as when you started.

5.2 TEST PLAN

This section is a technical description of the test plan that will be carried out on the Multi Domain Network Management (MDNM) configuration. The boundary device is a piece of the MDNM architecture. During this time the test plan will concentrate on the boundary device, its functionality, and security. This test plan will be followed under all testing circumstances.

The boundary device interprets specific protocols and through a set of rules, permits or denies the transmission of information. The transmission of information is what is being tested.

The high side network is managed by the central NMS running HP OpenView. The low side network is managed by the low NMS. The low NMS runs HP OpenView and acts as a collection station. It monitors its network and sends status information to the central NMS as requested. The central network management system manages all low NMSs to provide a network COP of all the available domains.

5.2.1 TEST PLAN 1

The MDNM engineering team has devised a series of tests designed for investigating the functionality of the boundary device. The test plan follows.

The boundary device is designed as a network traffic filter. It only allows particular SNMP and ICMP requests to pass through. All other requests should be denied. The first proposed test involves testing all possible SNMP traffic through an exhaustive testing plan. We will test both SNMP versions 1 and 2c.

SNMP VERSION 1

The commands that will be tested in SNMP version 1 are **set, get, getnext, trap (via snmptrap), trap (via snmpnotify)**, and **response.**

SNMP VERSION 2C

The commands that will be tested in SNMP version 2c are **set, get, getnext, getbulk, trap, inform**, and **response.**

5.2.1.1 TEST PROCEDURES

The testing will begin by systematically testing SNMP commands. One command will be tested fully while all other commands will be turned off. The flow control is managed by /etc/SnmpProxy.cfg. This file must be edited in order to test the boundary device. The file is broken into two sections and each section is divided into two columns. The sections are High to Low traffic and Low to High traffic. The first column lists the available SNMP commands and the second column allows or denies the flow of that particular SNMP command. One command should be tested at a time while the rest are turned off. This testing will continue until all commands have been exhausted. Results will be documented. The table below shows an example of the desired test results for a correctly configured boundary device. A test plan matrix is included in the Appendix A for use throughout the testing procedure.

Documentation is very important when creating test scenarios. Documentation must include:

- ❑ The test being performed.
- ❑ The current configurations of the network. Particularly the switches in /etc/SnmpProxy.cfg.
- ❑ The expected results.
- ❑ The actual results. (Does the proxy accomplish the required functionality for network management?)

Table 1. Example of Rules Seen with a Correctly Configured Boundary Device

COMMAND	DESCRIPTION	HIGH TO LOW	LOW TO HIGH
Set	Assigns a value to a variable	Conditionally permitted	Not permitted
Get	Returns value of variable(s)	Permitted	Not permitted
GetNext	Returns value list of variables	Permitted	Not permitted
GetBulk	Returns a number of variables	Permitted	Not permitted
Inform	Transmits unsolicited Information	Not permitted	Not permitted– Investigating
Trap	Transmits unsolicited Information	Not permitted	Permitted
Responses	Responses to commands	Not permitted	Permitted

5.2.1.2 TEST DEVICE

The MDNM engineering team has a program to ease the testing process of the boundary device. Our primary interface was written in tcl/tk and is called **snmptester.ui.tcl**. It works by accessing the snmp utilities provided with HP OpenView.

5.2.2 TEST PLAN 2

The second test plan will be to bombard the boundary device with some of the most current intrusion tools found on the Internet.

5.2.2.1 TEST PROCEDURES

The testing will begin by evaluating the available software for its functionality and usefulness towards hacking the boundary device. After software has been chosen, the security of the boundary device will be tested.

Documentation will follow the same guidelines of Test Plan 1. Documentation will also include the specific intrusion being launched towards the boundary device.

5.2.2.2 TEST DEVICES

The test devices used will be a compilation of intrusion tools collected by the MDNM engineering team.

5.2.2.3 TEST AREA

The test area will consist of the AFRL test network. The test network is represented in Exhibit 1 by the rightmost tree. We will be testing the boundary device that sits between the COP and the low network manager.

6.0 EXPERIMENTS

The Multi Domain Network Management (MDNM) engineering team participated in the Joint Expeditionary Force Experiment 2002 (JEFX) at Langley Air Force Base. This section describes JEFX, testing performed at JEFX, user feedback, experimental code development, and users' quotes.

6.1 JOINT EXPEDITIONARY FORCE EXPERIMENT (JEFX)

The JEFX is designed to assess Air Force operations in a simulated war-fighting environment. JEFX also helps promote Expeditionary Aerospace Force (EAF), the Air Force's new concept of operations. EAF allows a uniquely tailored force to be used in a specific operation by taking pieces of each division and deploying them individually. As an example, this selective deployment could be done in lieu of deploying the entire 9th Air Force.

The MDNM team was given a live SIPRnet feed from the Network Command Center Deployed (NCC-D) at Langley Air Force Base. The SIPRnet is a secret network to which the team was given access specifically for managing the 1,050 JEFX-related nodes. The boundary devices acted as controlled interfaces between the COP, the SIPRnet, and the team's model of the NIPRnet. The NIPRnet is an Air Force unclassified network. A model of it was used since the boundary device was not accredited and therefore did not permit spanning of multiple security domains. The team's demonstration provided a COP of both the JEFX SIPRnet and the modeled NIPRnet domains.

JEFX was held at various military installations across the United States. The main Air Force bases were Nellis in Nevada, Hurlburt in Florida, and Langley in Virginia. The air exercises took place at Nellis; the main operations took place at Hurlburt with Langley acting as a backup to Hurlburt. The MDNM team was stationed in the National Operations Security Center (NOSC) at Langley Air Force Base.

6.2 THE MDNM TEAM

The MDNM team was an integrated team composed of representatives from AFRL, BAE SYSTEMS, and ARCA Systems. Mr. Scott Shyne of the Air Force Research Laboratory Distributed Information Systems Division (AFRL/IFGA) managed this advanced technology demonstration (ATD) and was supported by Messrs. Joe Riolo and Adam Hovak of BAE SYSTEMS and Ken Seiss and Herb Markle of ARCA Systems.

Mr. Seiss was the developer for the MDNM project. Mr. Markle was the system architect. He also provided design assistance to Mr. Seiss and tested the code. Both Mr. Riolo and Mr. Hovak provided independent integration and testing of the boundary device as well as system and network administration functions. The team had great success working together.

6.3 TESTING

Testing is an essential part in developing a project. On the MDNM project, functionality, as well as security was tested. Functional testing was done to prove the MDNM accomplished the goals

it had been designed to accomplish. Security testing was done to expose vulnerabilities in the boundary device.

6.3.1 FUNCTIONAL TESTING

The team performed numerous functional tests at Langley Air Force Base.

6.3.1.1 TIMING TESTS

For each timing test the team gathered 12 synchronization times, discarded the high and low times, and averaged the 10 best.

In the first timing test the team tested the time it took for the COP to synchronize with a low NMS managing 250 nodes. The tests concluded that synchronization takes an average of 52.7 seconds to complete. During this synchronization, the Central Processing Unit (CPU) of the low NMS peeks at approximately 80%.

The second test consisted of a synchronization experiment between the COP and a low NMS managing 1,033 nodes. The team concluded that synchronization takes an average of 180.7 seconds.

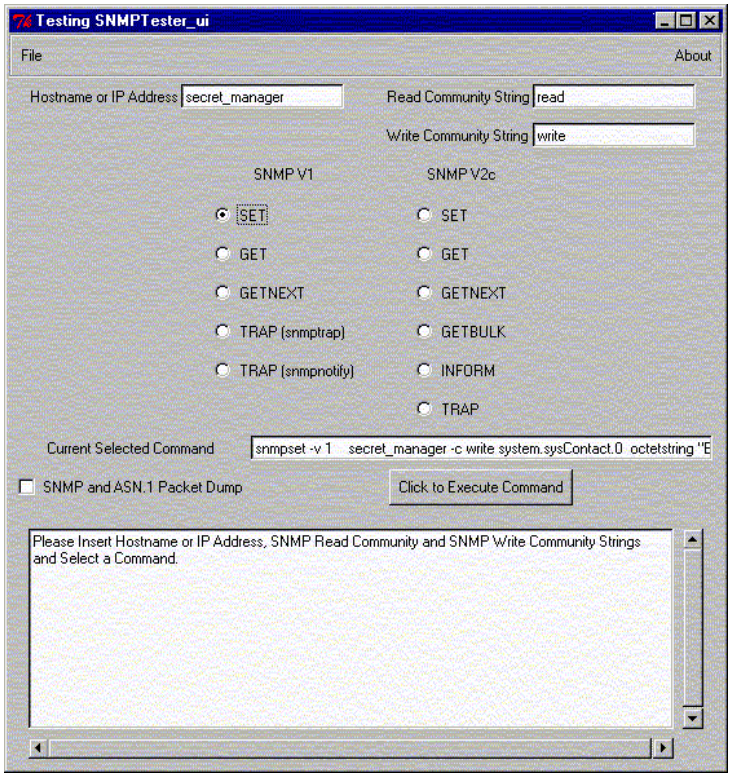
The third timing test consisted of 1,054 nodes. In this test it was found that it took an average of 185.1 seconds for the COP to synchronize with the low NMS. The team also noted that the CPU utilization on the boundary device was at approximately 57% while running x-windows and 27% while not running x-windows.

6.3.1.2 COMMAND TESTING

The third test was an exhaustive test whereby all SNMP commands were systematically tested.

The MDNM team created a program to ease the functionality testing process of the boundary device. The primary interface was written in tcl/tk and is called **snmptester.ui.tcl.** It worked by accessing the SNMP utilities provided with HP OpenView. A screenshot of snmptester.ui.tcl is shown below in Exhibit 2.

Exhibit 2. Screen Shot of snmptester.ui.tcl



One command was tested fully while all other commands were turned off. The flow control was managed by /etc/SnmpProxy.cfg. This file was edited in order to test the boundary device. It was broken into two sections and each section was divided into two columns. The sections were High to Low traffic and Low to High traffic. The first column listed the available SNMP commands and the second column allowed or denied the flow of that particular SNMP command. An example of /etc/SnmpProxy.cfg is shown below.

/* code snippet
SNMP v1 allow
SNMP v2 allow

H2Lset allow

L2Hset denied
L2Hget denied

***/ end of code snippet**

The testing continued until all commands were exhausted. Table 2 shows an example of the desired test results for a correctly configured boundary device.

Table 2. Rules a Correctly Configured Boundary Device Would Display

COMMAND	DESCRIPTION	HIGH TO LOW	LOW TO HIGH
Set	Assigns a value to a variable	Conditionally permitted	Not permitted
Get	Returns value of variable(s)	Permitted	Not permitted
GetNext	Returns a value list of variables	Permitted	Not permitted
GetBulk	Returns a number of variables	Permitted	Not permitted
Inform	Transmits unsolicited information	Not permitted	Not permitted (Investigating)
Trap	Transmits unsolicited information	Not permitted	Permitted
Responses	Responses to commands	Not permitted	Permitted

Documentation is very important when creating test scenarios. The documentation includes:

- ❑ The test being performed.
- ❑ The current configuration of the network, particularly the switches in /etc/SnmpProxy.cfg.
- ❑ The expected results.
- ❑ The actual results (e.g., Does the proxy accomplish the required functionality for network management?).

The functional testing proved to be fairly successful. Of 16 tests performed, only 2 failed. All commands were allowed through the boundary device except for SNMP traps. SNMP version 1 traps did not go through the boundary device regardless of the boundary device configuration. We gave this information to Mr. Seiss of ARCA Systems and it has been addressed.

6.3.2 SECURITY TESTING

During the security test plan the boundary device was bombarded with some of the most current intrusion tools found on the Internet.

The testing began with an evaluation of the functionality of the available software and its usefulness for hacking the boundary device. The tools chosen by the team were Ethereal, Strobe, Vetescan, and Ostronet. After testing each tool for ease of use and number of incorporated tools, the team chose Ostronet because it was easy to use as well as purposeful to the testing. Ostronet is a collection of Internet tools that provides a domain scanner, port scanner, Ping and Traceroute features, host address-to-name and name-to-address resolution, detailed output, Whois and Ph

clients, and a finger utility. The team chose this product because it was easy to use as well as purposeful in the testing.

6.3.2.1 PENETRATION TESTING

In the first test the team utilized the Scan Wizard featured in the Ostronet tools package. The Scan Wizard is a port scanner used to detect the boundary device. The scanner was assigned a subnet of 192.168.2.*; however, the scanner did not detect the boundary device in this test.

In the second test, the port scanner was used again to test all ports on the boundary device. After a thorough scan of the boundary device, no ports were shown to be open.

6.4 USER FEEDBACK

During the JEFX exercise the team interacted with many users in the NOSC at Langley Air Force Base. The users dealt with network management software everyday, managing both the unclassified NIPRnet and the secret network SIPRnet. The support team at Langley used programs like What's Up Gold and HP OpenView to manage their networks. One strong complaint about their software was that it was either too difficult to use or it was too general.

What's Up Gold is a network management program that shows generic network infrastructure. It only monitors specific switches and routers at critical throughput points in the network. The support group at the NOSC stated that What's Up Gold could detect a problem, but that it could not identify the problem. HP OpenView had to be used since this program has the ability to show which individual node has the problem. The support staff at the NOSC told us that HP OpenView is difficult to use because it provides a very complicated picture.

The support staff enjoyed the team's centralized view of the network and called it a "good idea." One member of the staff even asked why this idea took so long to be presented.

Another member of the staff mentioned that, "this capability is even more relevant now that 3-dimensional networking is available. 3-D networks allow multiple virtual LANs on the same network. This can allow different security domains to travel over the same infrastructure." For example, the JEFX Network infrastructure was classified as secret but with encryption, unclassified data could be sent over it. Therefore if a network administrator has both networks correlated on the same display it would be easier to pinpoint a network hardware problem.

6.4.1 SUGGESTIONS

Along with praise comes criticism. The team was lucky in that it received constructive criticism from actual network managers at JEFX. They said that a network common operational picture was very useful as long as some extra ideas were incorporated. They suggested adding a drill-

down capability, allowing the COP the ability to query specific system information from individual nodes. Adding a drill-down capability would facilitate the gathering of useful information such as system contact, system description, and system location. This information could then be used in debugging problems.

6.5 EXPERIMENTAL CODE DEVELOPMENT

The user feedback the MDNM team received from the support staff in the NOSC at Langley led to a new idea, which was to use the test network side of the MDNM demonstration to try out new code. This new code would be created during JEFX and it would provide the capabilities suggested by the NOSC support staff.

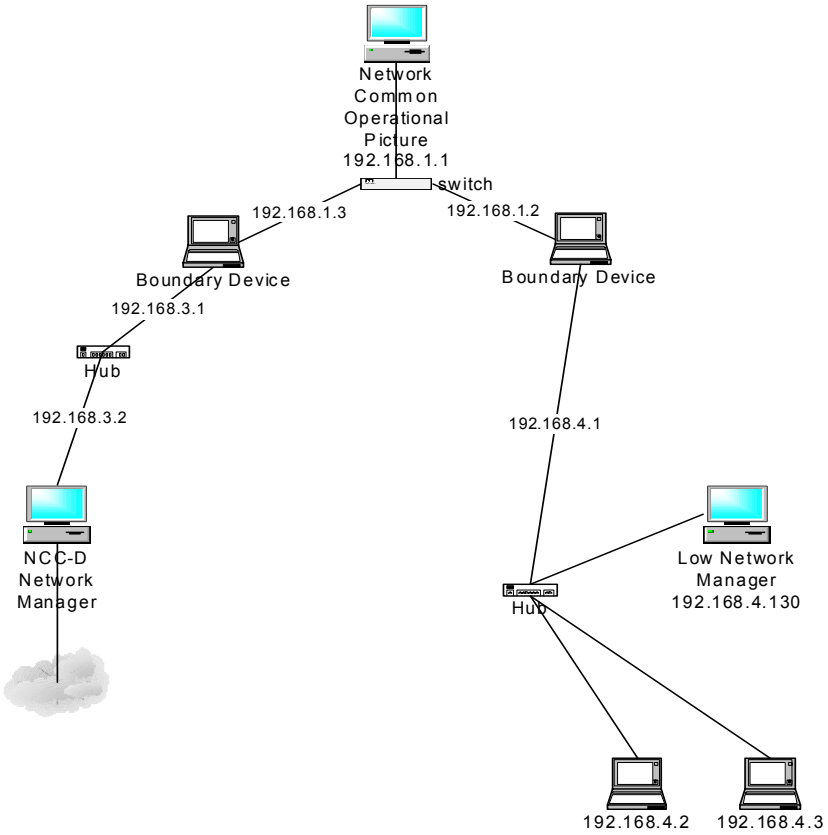
When Mr. Seiss arrived at the JEFX demonstration he was met with the challenge of creating new code to accommodate the suggestions of the NOSC support staff. The plan was to design a second proxy to run on the controlled interface. This second proxy would facilitate a drill-down capability to the COP, i.e., the COP would be able to query specific system information from individual nodes.

During the last few days of the exercise, an Alpha version of the second proxy was created. This proxy allowed the COP to query one node on the test network. The second proxy only allowed the COP to query one node because there were some troubles involving Access Control Lists (ACL). The Alpha version of the code was used to enhance the MDNM demonstration. The second proxy was run on the boundary device controlling the information exchange between the COP and the test network.

The second proxy, proxy2, was designed for a purpose different from that of the original proxy. In order for proxy2 to work the architecture of the demo needed to be changed (see Exhibit 3).

As seen in Exhibit 3, the boundary device is now connected directly to the lower domain. While this architecture allows the COP the ability to query individual hosts, it is also much more of a security risk. Adding the boundary device to the local domain allows the hosts direct access to it, and there is always a potential security threat if a user can access a boundary device.

Exhibit 3. New Multi-Domain Network Management Architecture



6.6 CONCLUSION OF JEFX

Upon the completion of JEFX, the MDNM team was met with a few challenges involving the removal of the demonstration hardware from the NOSC so it could be returned to AFRL. Since the equipment was connected to the secret network SIPRnet, the systems had to be declassified. Declassification involved "wiping clean" all the machines.

About two days before the exercise ended, the team began the tedious process of cleaning the hard disks. The team soon discovered that Langley support staff were not familiar with declassifying machines. In fact, they had no software suitable for this purpose.

A few phone calls were made to Air Force Research Lab (AFRL) asking for assistance. The site had two applications that could be used to declassify machines, UniShred Pro and Norton Wipe. The software was shipped overnight to the security manager at Langley. Once the team received the software, the first step was to start UniShred pro on each of the Sun boxes. It was not as easy, however, to declassify the Intel platforms. The program, Norton Wipe, was limited to wiping only 2-gigabyte partitions. In order to declassify the machines, the 2 20-gigabyte drives

had to be broken up into 10 partitions each. The other 2 hard drives were only 4 gigabytes and were broken up into 2 partitions each. After all the partitions had been created the team ran into one more problem: someone had to physically start each partition format every 90 minutes. This left the team with the inconvenience of manning the declassification process throughout the night. After a few scheduling discussions, a plan was decided and, by morning, all the hard drives were completely wiped.

The last task that was required before the team could leave the NOSC was the certification of the machine declassification. Lt. Warne was enlisted to supervise the removal of the SECRET stickers from the machines. Sgt. Porter checked out our equipment and the team headed home.

6.7 LESSONS LEARNED

JEFX was a very useful experiment in which to participate with the Multi Domain Network Management project. The team was afforded the chance to test the product in a real-world situation and receive user feedback. JEFX was structured such that the software could be tested in between demonstrations and new code created to reflect the user feedback received by the team. Having actual support staff give real-time input led the team to new development ideas as well as modifications to make the user's job a little easier.

6.8 IMPORTANT QUOTES

Throughout the two weeks at JEFX the team gave demonstrations to users as well as their supervisors. After the overwhelmingly positive feedback, it is useful to record the users' thoughts pertaining to the MDNM project.

"This type of system would give me the capability to know specifically what systems within both SIPRnet and NIPRnet were operational from one workstation." Captain Hugh St. Martin

"What took so long? Why didn't somebody think of this earlier?" Captain Matthew Gebhardt

"This fine grain control may significantly enhance firewall capability at the base and MAJCOM level." Captain Travis Ross.

"The correlation capability provided by this system would give us a much better understanding of the potential relationships of individual network events occurring within different security domains." Tech Sergeant Terrence Bruce

"I hope you're planning to participate in JEFX '02 as a Cat I or Cat II. I believe your program is providing a critical capability to the Air Force Network Operations Personnel." Lieutenant Colonel Gordon Mann

7.0 CONCLUSION

This document has sought to provide an in-depth description of the Multi-Domain Network Management boundary device. It has provided the steps necessary to set up the boundary device.

APPENDIX A

TEST PLAN 1.0

Tester: _____ Date: _____ Time: _____

Description of Test:

Current Configuration (including information flow between network managers):

Test Matrix (use allow (allowed) and deny (denied))

SNMP Command	Allow	Deny	Test Results
set (v1)			
get (v1)			
getnext (v1)			
trap (v1 via snmptrap)			
trap (v1 via snmpnotify)			
response (v1)			
set (v2c)			
get (v2c)			
getnext (v2c)			
getbulk (v2c)			
trap (v2c)			
inform (v2c)			
response (v2c)			

Results Summary

Test Score: _____ Pass _____ Fail _____ Retest